Machine Learning-based Classification System for Building Information Models

Minjung Ryu

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 24.02.2020

Supervisor

Prof. Hong-Linh Truong

Advisors

M.Sc. Matti Kannala

M.Sc. Cornelius Preidel

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Minjung Ryu

Title Machine Learning-based Classification System for Building Information Models

Degree programme Computer, Communication and Information Sciences

Major Computer Science **Code of major** SCI3042

Supervisor Prof. Hong-Linh Truong

Advisors M.Sc. Matti Kannala, M.Sc. Cornelius Preidel

Date 24.02.2020 **Number of pages** 51 **Language** English

Abstract

Building Information Modeling (BIM) provides information regarding a building in digital format for the comprehensive building life cycle, which comprises building design, construction, and operation. Compared to traditional manual processes, which are based mainly on 2D drawings, BIM has a huge potential to reduce time and costs, and increase efficiency. The concept of classifications in BIM is particularly crucial throughout the life cycle of BIM since it gives structure to the data and helps users analyze different parts of the building model. Despite its importance, only a few BIM tools offer accurate handling of classifications but still with great limitations. They require human expertise and effort for manually assigning the right categories or defining classification rules for assigning the categories. The main objective of this thesis is to improve upon the conventional and largely manual approach to element classifications by utilizing rich data in BIM. To achieve this, three different types of information, which are numeric sizes, textual descriptions, and 3D shape, are extracted from elements in the BIM model. To process this information, a classification model, which concatenates outputs from a convolutional neural network and a fully connected neural network, is trained. Furthermore, the study defines metrics for measuring the quality of the data and examines their relationship with system performance. Moreover, the prototype of the classification system is deployed to a cloud environment to investigate the impact of system configuration on the performance. The results of the experiments empirically prove that a data-driven approach using the rich information of BIM enables automatic classification using machine learning. Experiments on different system configurations show that additional computing resources improve processing time with a trade-off of cost. Analysis of the quality of data and time-accuracy trade-off offers an insight into the optimal data selection and system configuration with practical examples. Finally, this study illustrates the potential of machine learning for improving the BIM classification tools that can be integrated into existing BIM tools.

Keywords machine learning, object classification, BIM, IFC, 3D convolutional neural network

Acknowledgements

I would like to express my deepest appreciation to my supervisor, Professor Hong-Linh Truong, for his insightful guidance and encouragement.

I am also indebted to Solibri for giving me the opportunity to work on this interesting topic. I am grateful to all my team members, and advisors Matti Kannala and Cornelius Preidel. Thank you all for the helpful comments and inspiring discussions.

Most of all, I wish to thank my lovely family for always being my biggest supporters.

Espoo, February 24, 2020

Minjung Ryu

Contents

Abbreviations

1D	1-Dimensional
2D	2-Dimensional
3D	3-Dimensional
AEC	Architecture Engineering Construction
AWS	Amazon Web Services
BIM	Building Information Modeling
CNN	Convolutional Neural Network
CPU	Central Processing Unit
EC2	Elastic Compute Cloud
FM	Facility Management
FN	False Negatives
FP	False Positives
GPU	Graphics Processing Unit
IDF	Inverse Document Frequency
IFC	Industry Foundation Classes
ITO	Information Takeoff
kNN	k-Nearest Neighbors
ML	Machine Learning
MLP	Multilayer Perceptron
ReLU	Rectified Nonlinearity Unit
S3	Simple Storage Service
SMC	Solibri Model Checker
SQS	Simple Queue Service
SVM	Support Vector Machines
TF	Term Frequency
TP	True Positives
vCPU	Virtual Central Processing Unit
X^2	Chi-square

1 Introduction

In the architecture, engineering, and construction (AEC) industry, Building Information Modeling (BIM) is a key element for the digital transformation of the industry. It means the use of digital information on a building model over the course of the entire building life cycle including design, construction, and operation phases [1]. The digital representation of a building model offers a framework for communication and collaboration with consistent and coordinated information to all project participants, such as owners, architects, engineers, and constructors. As a result, it reduces time, cost, and errors, and increases efficiency compared to the conventional manual processes [2].

As all the information about a building is centralized and integrated, effective ways of managing the increased amount of data have emerged as a critical issue in BIM. Of all the features of BIM, this thesis will focus on classification since it is a crucial tool to help the users keep control over the vast amounts of data that typically accumulate in a construction project. Classification is used in BIM to identify and organize building elements by grouping them into multiple categories and adding labels to them based on specific matching characteristics. This helps stakeholders manage, coordinate, and evaluate BIM models across disciplines by making it easier to find and select appropriate building elements. As a result, this supports standardized identification that is further used for project management, scheduling, cost estimation, and quality assurance. Besides, the classification enables the stakeholders to understand the structure of BIM information even without manipulating the actual data. Various classification standards for building elements have been developed and updated with the help of digital information and tools as the available information expands in the building industry. For example, MasterFormat, Uniformat, Uniclass, and OmniClass are the most popular ones that are widely used all around the world [3].

In digital classification tools, there are three major approaches of assigning category labels to building elements: information reuse, manual assignment, and automatic assignment. Information reuse is when the building elements are copied from an old BIM project of the previous work and used again for a new BIM project. In this case, the elements keep the previously defined information as properties of building elements including the category labels that are assigned in the old project. Manual assignment can be done by many BIM tools such as ArchiCAD and Revit. They are two popular examples of BIM authoring software that create the actual BIM models. These software products provide classification tools: ArchiCAD Classification Manager [4] and Revit Classification Manager [3]. Their classification tools allow users to map selected elements to the desired category from pre-defined or custom classification database. However, these tools require manual assignment, meaning that users need to select the element and target category label to classify. Automatic assignment is provided by Solibri software [5] which is a software tool that analyzes the quality of BIM models. It finds, visualizes and reports problematic issues in a building model for quality assurance. The classification in Solibri software works with an improved approach based on rules on specific attribute fields of each element. This

tool automatically selects the target elements that satisfy the classification rule and adds a category label to the elements according to the rule definition. While detailed instructions for adding the classification rules are given in the online manual of the Solibri software product [6], an example of rule definitions in the Solibri software is shown in Figure 1. In this example, a classification rule for the category *Exterior Walls* searches all the elements whose *Component* field is *Wall*, or the *Type* property contains both *Basic Wall* and *Exterior*. The processes of selecting target elements and assigning them to corresponding categories are automated in Solibri software. However, the classification rules still need to be defined by users.

1.1 Problem statement

Throughout the entire stages of design, construction, and operation in BIM, classification is used extensively for grouping and filtering the building elements. A central problem, however, is that existing classification tools require either direct category assignment or classification rule definition from human input. Despite the time-consuming manual work, the result of manual classification does not always exhaustively assign all elements to the corresponding categories and may leave some elements unclassified or misclassified due to different ways of modeling, slight deviations, or errors. There are several risk factors in manual classification:

- Misuse of data: Elements are not used as planned due to users' misunderstanding

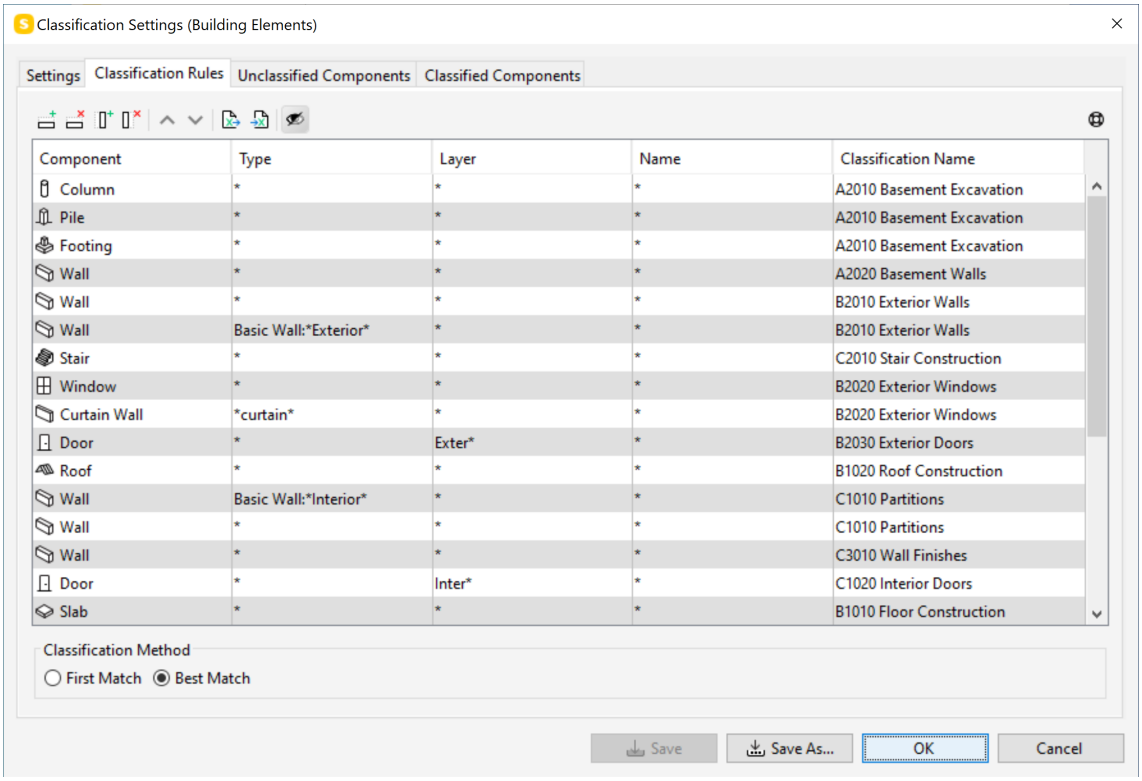


Figure 1: Classification rule settings in Solibri software [6]

about classification. For example, ceiling elements are used to represent floor elements because they have similar shapes and properties.

- Inaccurate data: Users could make a mistake when they manually fill the value of fields, such as from typos. Often customers download the BIM elements from an internet website or copy elements from their old projects, and the reused elements may still contain unintended values in certain fields that are not related to the new project.

- Missing data: If an attribute field for a code is missing, it cannot be automatically classified by defined rules at all, unless a user defines all possible attributes that could have the target value.

Classification information is used not only to track down the target information but also on other phases of BIM, such as structural analysis, cost estimation and quality assurance. Therefore, the weaknesses associated with manual classification can also negatively influence those stages. In structural analysis, engineers extract only structural elements such as floors and columns from BIM to evaluate the building structure. Cost estimation quantifies building elements in each category, and it calculates the cost based on the result of classification. Quality assurance examines problematic issues or requirements that the model should satisfy. Classification is done before issue checking so the category can be used as a filter. For instance, when a project requires all exit doors to operate in the correct direction, classification can be applied to check only on the doors in the exit door category. If the classification is incomplete or inaccurate, the issue checking cannot detect object clashing or cost estimation can erroneously exclude certain elements. Hence, this study investigates the problem of manual classification and suggests a solution to improve the current classification tools.

1.2 Goals

The aim of this thesis is to design the architecture for a BIM element classification system which does not require human intervention. In order to develop this solution, the thesis will use a data-driven approach utilizing the rich amount of data in BIM as shown in Figure 2. This will be accomplished by extracting relevant data from the BIM model and applying suitable machine learning algorithms on the data. Finally, the prototype of the system will be implemented to recommend its applicability to practical system environments.

1.3 Scope

The thesis is limited to handle specific BIM file formats: Industry Foundation Classes (IFC) and Solibri Model Checker (SMC) format. IFC is one of the most used standard formats of exchanging digital building data. It is an open international standard (ISO 16739-1:2018) developed by buildingSMART for BIM interoperability [7]. SMC is a BIM model format for Solibri software that derives valuable data in addition to

the original IFC file for quality assurance without changing the actual IFC data. The detailed implementation of the data collection component in the system is excluded from the scope of the thesis since this component is part of Solibri software.

1.4 Contribution of the thesis

The proposed classification system performs two major tasks: data collection and machine learning. The data collection task determines the relevant and important data about building elements within the BIM model. The machine learning task automatically creates a classification rule based on the extracted data and applies the rule to the unclassified elements. This machine learning-based classification system complements existing automatic classification tools and helps users save time so they can focus on other meaningful tasks instead of the tedious manual work. There have been several pieces of research that apply machine learning methods to BIM element classification, however, most of them are focused narrowly on only machine learning tasks considering few types of building elements from a couple of simple BIM models. On the contrary, this thesis studies BIM element classification in an end-to-end manner from data collection to prototype deployment to different computing environments. Also, many different BIM models and their individual classifications are used to evaluate the performance of the system. Furthermore, this thesis shows that the classification system performance depends on the data quality of BIM and finds out the accuracy-cost trade-offs in the system configuration and resources.

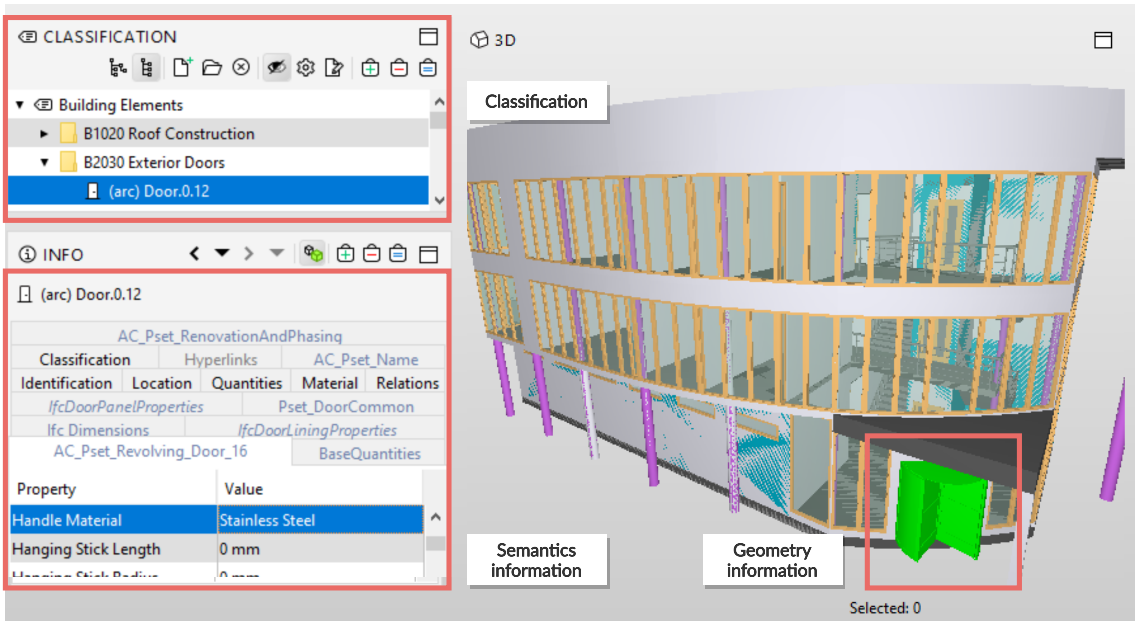


Figure 2: BIM data in an IFC file visualized by Solibri software

1.5 Thesis structure

This thesis is organized as follows. Section 2 provides a brief introduction to the previous studies on machine learning regarding BIM element classification as well as theoretical background of the overall approach to this research. Section 3 presents an overview of the proposed classification system architecture and detailed approach of each system component. Section 4 describes the implementation details of the prototype system. Section 5 presents experiment settings to evaluate the system and discusses potential factors that influence system performance. As a result of the experiments, trade-offs between data quality, system configuration, and system performance are analyzed in Section 6. Finally, Section 7 concludes the research by summarizing the system performance and data quality analysis and suggests possible future improvements in the system.

2 Related Work

This section introduces literature on the issue of classification. This section first summarizes and explains classification in the perspective of machine learning in Section 2.1, and then, provides a general review of the state-of-the-art research into 3-dimensional (3D) object classification using machine learning approaches in Section 2.2. Section 2.3 presents several specific approaches to identifying building elements in BIM models using machine learning.

2.1 Machine learning-based classification

Machine learning (ML) can be defined in multiple ways. One of the simple definitions is that ML is to program computers to optimize parameters to predict or describe a model using data from past experience [8]. Although there are many definitions of ML, the fundamental idea of ML is to learn patterns from some observed data in order to solve a complex problem without explicitly defined rules. There are many advantages of using ML in classification problems [9]. ML classification provides an automatic method, which is driven by data, instead of by human experts. And it is often more accurate than human-crafted rules.

The two main types of ML are supervised learning and unsupervised learning [10]. Supervised learning finds the mapping from input to output from data that are associated with correct labels provided by a supervisor. In other words, this requires the data to be a pair of input and output labels and finds a function that infers the output label for the input. Classification and regression are examples of supervised learning, which are designed to predict the correct answer of categorical output or continuous numeric output. Unsupervised learning only comes with input data. Instead of predicting the output variable for unseen inputs, unsupervised learning tries to find the structure in, or distribution of, the given data. For instance, clustering places similar examples from the input data into groups.

Classification, one instance of a supervised learning problem, is one of the most frequent machine learning problems in which many different authors have developed algorithms to solve the classification problems, as reviewed in [11]. In [11], the key algorithms for classification problems are reviewed and compared. For example, decision trees and rule-based classifiers are given as logic-based algorithms, and multilayer perceptrons (MLPs) are explained as one of the perceptron-based techniques. Additionally, Naive Bayes classifiers in statistical learning algorithms, k-nearest neighbors (kNN) in instance-based learning, and support vector machines (SVM) are introduced. In this thesis, Convolutional Neural Network (CNN), which is inspired by MLP, is applied to building element classification problem. CNN is one of the most popular algorithms for solving 3D object classification problems. This is explained in Subsection 2.2 in detail.

2.2 3D object classification

This subsection explains CNNs, a specialized class of multilayer perceptrons (MLPs). MLPs, which are also called feedforward neural networks, consist of layers of neurons whose structure is designed to imitate human brains structures. Training in algorithms using feedforward neural networks with multiple layers is called *Deep Learning*. Deep learning automatically learns the representations of data from different layers of abstraction due to the deep network structure [12]. CNNs process data with a known grid-like topology [10, 13], thus images that can be represented as a 2-dimensional (2D) grid of pixels have been successfully analyzed by CNNs [10]. One topic of the image analysis field in deep learning is image classification. Recent state-of-the-art techniques for object classification in 2D images are considerably improved by CNNs along with larger datasets and powerful Graphics Processor Units (GPUs).

As 3D shape representation has become more prevalent due to the availability of inexpensive depth-sensing cameras, the approach for 2D CNNs has also been applied to classify 3D objects in many computer vision publications [14, 15] using a grid of 3D voxels instead of 2D pixels. The most popular dataset for 3D object classification is ModelNet10 and ModelNet40 datasets, which contain 10 and 40 categories of general objects such as chair, bed, bathtub, and wall to name a few. This method that extends the approach of 2D CNNs is called volumetric CNNs. Another popular approach for 3D image recognition is multi-view CNNs. Multi-view CNNs aggregate the result of 2D projections of 3D shape from different angles, and can thus leverage the advances in 2D object classification and benefit from the complicated structure of neural networks that are already trained from a large scale image database such as ImageNet [16, 17]. For this reason, current state-of-the-art multi-view CNNs outperform volumetric CNNs in accuracy [18]. However, the set of 2D projections in multi-view CNNs cannot represent the 3D shape fully as some detailed information (e.g., curvatures) can be lost during the conversion from 3D to 2D shape [19]. For this reason, volumetric CNNs are used in this thesis for classification system implementation and the algorithms that are introduced in [15] will be reviewed in Section 3.

2.3 BIM element classification

This thesis will focus on 3D object classification in the specific field of BIM. As introduced in Section 1, there exist tools for manual BIM element classification. However, automated BIM element classification including machine learning applications have been researched to support BIM interoperability and detect misclassifications of IFC entity types (IFC classes). To automate the BIM element classification, several approaches were studied. One approach is using hand-crafted 3D shape features. [20] proposed a rule-based algorithm to classify IFC objects into predefined categories using geometric properties. They preliminarily experimented to find cone frustum-shaped piers. To do this, three mathematical definitions of cone frustum shape were identified: 1) There are two and only two circle faces, 2) The two faces are in parallel, 3) A line that connects the centers of the two faces is perpendicular to the faces.

The experiment was done with three elements, however, the scope of the experiment was limited to the cone frustum shaped objects and manual rule definition for each different shape of the object are required. Therefore, it is challenging to define the mathematical rules because it requires human expertise and significant time as there are many objects to identify in building models. In addition, machine learning-based approaches were also studied in [21] and [22]. [21] presented a machine learning approach to anomaly detection for miscategorized wall elements in a single BIM model using gyradius, volume, and area information. [22] extended the study of the machine learning method in [21], using a Support Vector Machine (SVM), to classify IFC classes based on geometric features including width, height, length, orientation, area, volume, and gyration. They experimented on four IFC classes: *ifcWallStandard-Case* and *ifcDoor* entities from three architectural models, and *ifcWallStandardCase* and *ifcColumn* from six infrastructure models. Deep learning and CNNs have been applied to classification problems in BIM data before, but the application has been limited to specific object types or non-object level categories. [23] studied using multi-view CNN methods to classify indoor point clouds into office furniture objects for a web-based visualization application in the context of Facility Management (FM) [24]. [25] applied 2D CNNs to classify building structure into one of three categories: Apartment building, industrial building or other.

Limitations of previous works on BIM element classification are that the experiments were done targeting only a few specific categories from a small number of BIM models. Besides, the experiments require a manual rule definition. In most cases, users define their custom classifications either based on IFC classes or international classification standards, such as Uniformat [26] or Omniclass [27]. According to the 2821 classifications from 414 model files that were used in the study, 60% of classifications are custom classifications that do not follow any standard. This means that each classification case has its own rules that need to work independently. Therefore, this thesis presents a system that utilizes big data and automatically learns the custom rules that users defined. In this way, the system supports all different kinds of classification rules flexibly, instead of only some specific rules or standards. The proposed classification system in this thesis uses the characteristics of each element. This approach is similar to the existing studies that are introduced in Section 1 as the previous works also considered element sizes and descriptions as input data. However, the difference is that the proposed system does not necessarily require the rules to be predefined such as specific code or shape features. In this thesis, volumetric CNNs using VoxNet [15] will serve as the central method because it offers the advantage that this method can fully represent the actual 3D shape. Numeric features for the network will be globally defined from size information to the location of the object. For textual information, it is enough if the code or specific keywords exist in any form in it, because the suggested method considers all the textual properties.

3 Architecture and Functionality

This section will introduce the architecture and functionalities of the system that is implemented for this study. The proposed ML-based classification system in this thesis consists of five major components: data collection, data pre-processing, feature selection, ML model training, and inference (prediction). An overview of the architecture is shown in Figure 3. In this study, the classification system has two alternative configurations, as shown in Figure 3.

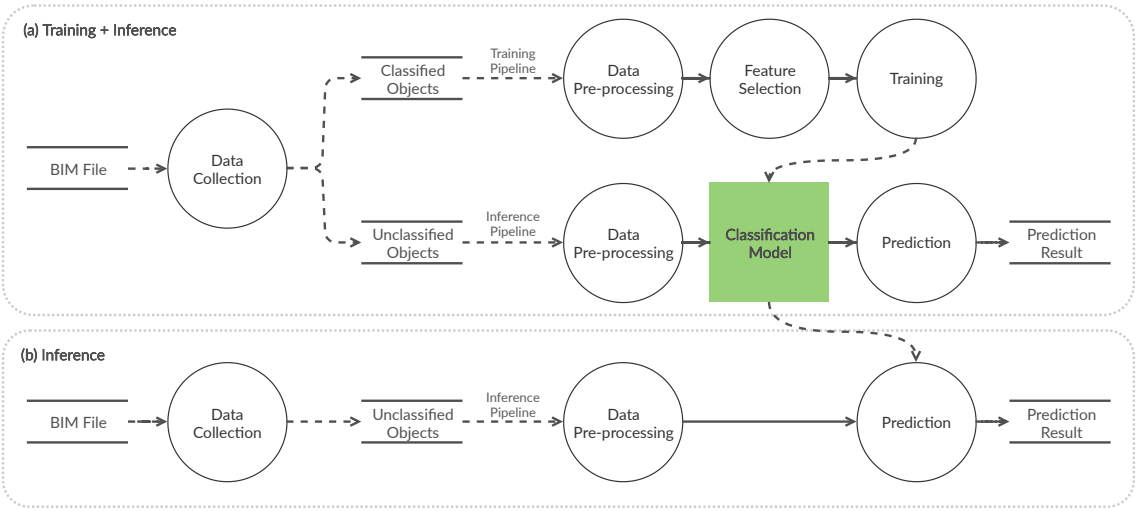


Figure 3: System Pipeline

Figure 3 (a) shows the configuration in which classification training and inference happen in a sequence. Figure 3 (b) shows the configuration with an inference phase only. The training phase is not needed here since a pre-trained classification model is available to reuse. Two configurations were devised for the experimental classification of BIM elements in two different situations.

1. Situation 1 (Configuration 1): Train a neural network based on the classified elements in a BIM file. Then, predict the unclassified elements in the same file. This is useful when unclassified building elements remain after manual or rule-based classification. It also improves upon the classification rate of the existing classification tools.

2. Situation 2 (Configuration 2): Train a neural network based on the classified elements in one BIM file. Then, make a prediction using the classifier on all the elements in other BIM files. This is more useful than Situation 1 when users apply the same classification rules to multiple BIM models. This configuration can also detect misclassified elements by comparing the actual category and the predicted category.

Both configurations assign correct categories for unclassified elements. However, the difference is that in Configuration 1, training and inference are done in one

model file while configuration 2 can be applied to multiple files. Once there is a trained model, it does not need to be re-trained to classify the elements in a new file. Re-training only needs to be done when new data requires a new category to be added to the classification system.

3.1 Data collection

The data collection component receives IFC files and SMC files as source data. IFC files consist of semantic descriptions and geometric representations. The semantic information contains characteristics of building elements that are defined according to the IFC schema. The semantic information is dynamically and externally added by users as attributes of building elements. This semantic information is usually in the format of a number, a Boolean truth value, and text. Along with semantic descriptions, geometric information is represented by surfaces and volumetric solids [1]. SMC is a native format of the Solibri software product [5]. SMC files contain exported information from IFC files with further information about building models that are either processed by Solibri software or defined by users. This rich information that BIM models provide makes it possible to describe and identify elements. In particular, the data collection component will extract the numeric attributes (dimensions and location), textual attributes (descriptions), and 3D attributes (shape) of BIM elements.

3.1.1 Numeric attributes

Each BIM element has many single numeric attributes that represent its characteristics such as size, weight, amount, count, and index. These numeric values are either continuous or discrete numbers that can distinguish one element from another. They are naturally passed to the training component as input data for ML algorithms.

3.1.2 Textual attributes

As with numeric values, string values are used to describe the properties of the building elements. All the string properties are extracted from an element even though not all other elements have the properties. In this way, the training can use all the textual properties to classify the elements.

3.1.3 3D attributes

Often a building element contains a description of its type or name as textual properties. However, sometimes the element has insufficient information to be classified into the correct category. In this case, the only reliable information of the building element is geometric information that is represented as a 3D object in BIM. From this geometric information, single numeric properties such as size and volume can be calculated. However, small details that tell about the shape of the element are not easily converted to a single numerical value. For example, a cube with a height of 10 cm and a sphere with a diameter of 10 cm have the same bounding box height,

width, and length in numeric size attributes although they are different objects. To deal with this difficulty, the proposed classification system utilizes 3D shape information in addition to the size information. 3D mesh information is represented as a set of voxels on the occupancy grid as experimented in the work of Wu et al. [14]. Wu et al. split the object area into $24 \times 24 \times 24$ cubes since this would have the same size of information as the high-resolution 2D 165×165 images [14]. In this thesis, different resolutions of 3D array will be experimented with in addition to $24 \times 24 \times 24$ resolution. If the cube overlaps the object, the tensor element has value 1 (true) in the corresponding location in the grid. The information is used as an input variable for the 3D deep neural network. Since 3D convolutional neural networks can use a 3D array as a training data, the geometric shape is extracted as a simplified representation (a 3D voxel grid as in Figure 4). This consists of $24 \times 24 \times 24$ size of binary variables that indicate the occupancy of an object in a 3D space [14, 15].

Before calculating the occupancy grid, the object is first transformed to be object-oriented. In this way, all the objects are aligned in a cube in local coordinates independently of how the object is placed in the global space. Then, the voxel occupancy is set to 1 if the object occupies the voxel, or 0 otherwise.

3.2 Data pre-processing

To improve the performance of ML training, the collected data is pre-processed beforehand. Data pre-processing includes data cleansing and data transformation.

3.2.1 Data cleansing

The data pre-processing component performs data cleansing to make sure the object does not contain invalid or useless information. For example, the ID of an object is not passed to ML training since it is unique for each element in a BIM file. Also, if the numeric attributes or geometric attributes are missing, the data is automatically filled with 0 so that the training can be done based on the other available attributes.

3.2.2 Data transformation

The collected data is also transformed to improve the performance of the system and converted into appropriate formats for neural networks.

Unit conversion All length properties are converted to mm, area to m^2 and volume to m^2. Numeric values that are in string format are converted according to the standard units above.

Standardization All numeric values are standardized to have mean 0 and standard deviation 1.

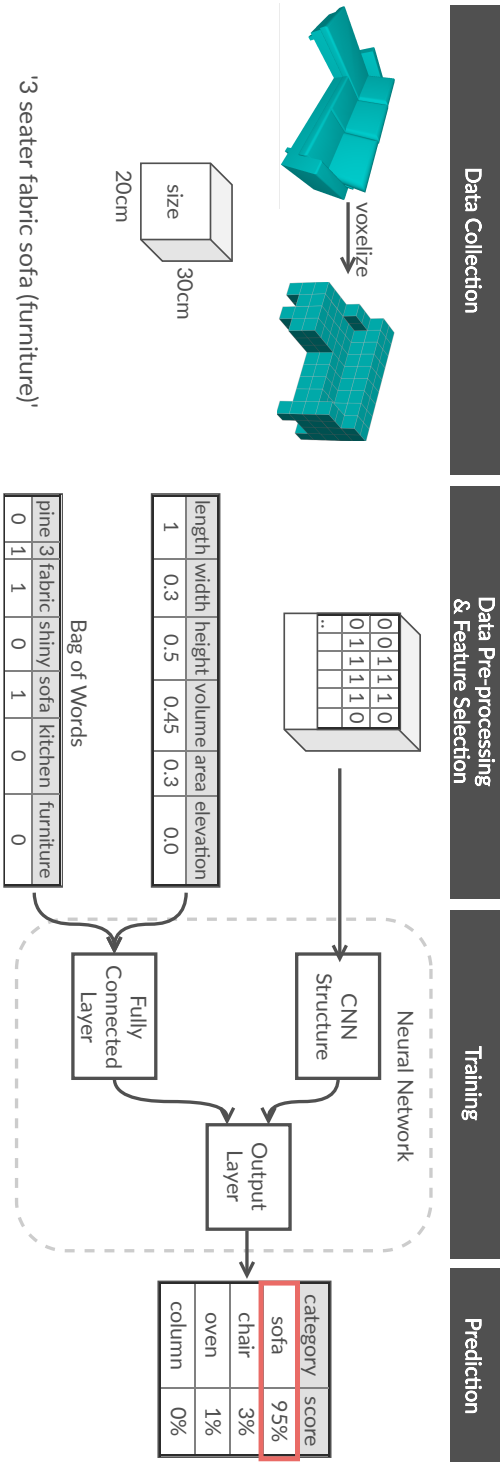


Figure 4: Data Pipeline

Text tokenization All texts are split into tokens. The tokens are changed to lower case. Stemming or stop-word ignoring are not done here since most of the textual values are independent basic-form words that are not in sentences. However, unique IDs and special characters are removed from the texts.

Text vectorization For text input data, similar string values can be under many different keys, and the applied machine learning algorithm will focus on the occurrence of each string value, not their location or name of the keys. The conventional automatic classification tool in Solibri software product is based on rules that check textual values in specific properties. Therefore, users need to specify which property to look up in order to classify the building element. For example, to find *Wall* elements, the user would create a rule that checks if the property *Name* contains *Wall*. However, this rule-based classification has the downside that the property name is not always in a fixed location. Depending on the models, the property to look up can be *Property name* or *Short name*. In this case, the rule cannot find the keyword *Wall* from the pre-defined property *Name*. It is possible to add more rules that check other properties. However, this still does not guarantee that all related properties are checked. In this case, important keywords about the elements are described as a text value, but these keywords are scattered everywhere in different properties. To gather this information, all the terms should be combined into one document form. In other words, the document is a collection of the terms, and this is transformed and represented as a bag-of-words model [28]. The bag-of-words model is a common method for representing the frequency of each keyword in a document.

As machine learning algorithms cannot use the raw description of element properties, these text descriptions need to be transformed into numeric values. The bag-of-words model transforms the document (in this case, list of words) into simple word-number pairs. The number is the frequency of the words, which indicates how many times the word appears in the document. Using only the occurrence, the ordering of the terms is ignored. Since the documents used in this thesis are originally formed by simple concatenations of words, and therefore lack positional information, the bag-of-words model is suitable for this case.

Finally, all property information in numeric and textual formats is transformed into numeric values so that the proposed classification system can use the data for training. Geometry input data is already in the numeric format as a 3D array, therefore the data pre-processing component does not transform this data.

3.3 Feature selection

Attributes are often called features in ML. BIM models can have a huge amount of properties and all properties are not related to the classification task. Therefore, to reduce the dimension of feature space and improve the efficiency and performance of training, not all attributes are used as input data. Using selected few features also brings generalization and avoids problematic overfitting. Overfitting is a type of error in ML training, where the machine learning algorithms fit the data into a too complex model. The overfitted model cannot generalize the input and fails at

predicting the result unless the input is the same as training examples. To prevent overfitting, the system will take a limited number of numeric and textual properties which are useful to identify the category of each BIM element.

One BIM element can have many different numeric properties, however, not all elements have the same set of properties. If all BIM elements do not have the same properties, ML training cannot be applied. Among other numeric properties that BIM elements have, some size information can be extracted from the geometry representation and all elements will have values for the properties. Therefore, the bounding box size (length, width, height, volume, bottom area, and bottom elevation) is calculated and extracted from the geometry.

The bag-of-words model which contains the occurrence frequency of each term in each category has been generated in the data pre-processing phase. To find the important keywords that characterize the document well and predict the correct category, the dictionary (collection of 'keywords') needs to be created. Later, only the selected keywords from the dictionary will remain in the bag-of-words model so that the terms that cannot distinguish documents are not included as input data. Only a few important term features are selected using two vocabulary feature selection techniques, term frequency thresholding which is a simple technique for vocabulary reducing, and Chi-Square test which is one of the feature selection methods [29]. Before the Chi-square test is applied, term frequency thresholding will be done since it will reduce the number of vocabulary and make the expensive Chi-square test be performed on fewer data.

3.3.1 Term frequency thresholding

To quantify the importance of the term, the concept of term frequency–inverse document frequency (TF-IDF) [28] in information retrieval is used. TF-IDF measures the importance of each term by counting how frequently the term occurs in a document or all documents of the category.

TF If the term is mentioned many times in a category, this term is important. If the term is referred to in only one document where there are hundreds of documents in the category, it means the term does not play an important role in classifying the category. Equation (1) describes how TF value is calculated for category c.

$$TF(c) = \frac{\text{number of term occurrence in category } c}{\text{total number of documents in category } c} \tag{1}$$

From this concept, terms that are used in less than 20% of all the documents in the same category will be removed from the bag-of-words model.

IDF If the term is used only in some specific categories of document, the term is important. If the term is used everywhere, this term cannot be used to classify documents. The original IDF for term t is calculated as Equation (2).

$$IDF(t) = \log \frac{\text{total number of categories}}{\text{number of categories where } t \text{ is mentioned}} \tag{2}$$

From this concept, terms that are used in more than 90% of all the categories will be removed from the bag-of-words.

3.3.2 Chi-square test

Chi-square test, also known as X^2 test is applied for feature selection. It measures the dependence between two variables, each feature, and categories [28]. For example, terms with 0 Chi-square value are not considered since it means the term and the category are independent. The importance of a term t in category c is defined as below, where A: the number of times t and c co-occur, B: the number of times the t occurs without c, C: the number of times the c occurs without t, D: the number of times neither c nor t occurs, N: the total number of documents:

$$X^2(t,c) = \frac{N \times (AD - CB)^2}{(A+C) \times (B+D) \times (A+B) \times (C+D)} \tag{3}$$

Then, $X^2(t)$ is calculated as Equation 4 to measure the importance of each term for total k categories.

$$X^2(t) = \sum_{n=1}^{k} X^2(t,c) \tag{4}$$

From this result, terms with the top 200 X^2 values are selected. If there are less than 200 terms after thresholding, all terms without 0 chi-square value are selected.

3.4 Training

A neural network model is adopted to construct a classifier for each classification rule in a building model. This machine-trained classifier will learn the category mapping rules from the BIM elements that are already classified by users and pre-defined rules. The architecture of the neural network has two branches for two different types of inputs: 1-dimensional (1D) array which contains numeric values and bag-of-words values, and 3D array for geometric occupancy grid. A CNN branch for 3D data and a MLP branch for 1D numeric data are combined for the mixed types of input data. In this thesis, the structure of VoxNet [15] is adopted and used as the CNN branch. It is one of the earliest studies in volumetric CNNs for 3D object classification and also showed the significant success of 3D CNN with a comparatively basic structure [30]. The implemented neural network structure for combining the 3D shape, bag-of-words representation and numeric properties are presented in Figure 5.

3.4.1 Input layer

The input layer takes two different types of data: a 3D shape and numeric data which consist of size measure and text information. The original 3D shape information is represented as a $24 \times 24 \times 24$ 3D array as explained in Section 3.1 and 4 extra voxels are added to the input data as a margin since CNN layers in VoxNet receive $32 \times 32 \times 32$ 3D arrays. The length of element size information is fixed as 7 and the

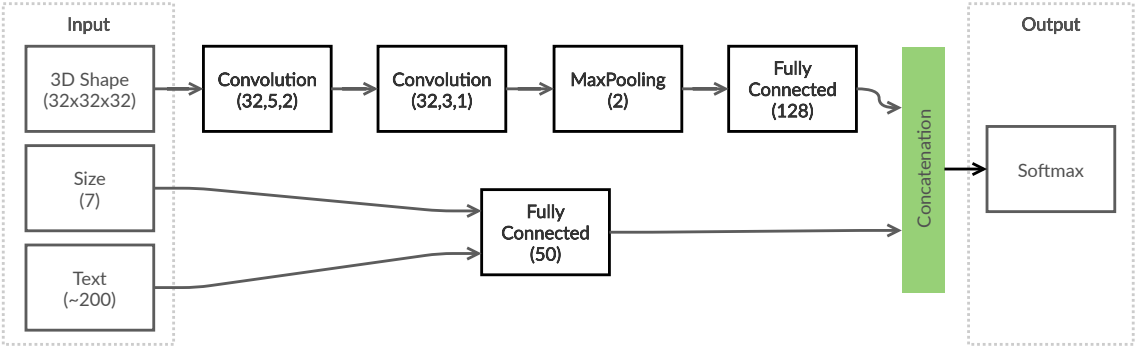


Figure 5: Structure of neural network in the pipeline

length of the term occurrences is maximum 200. The occupancy grid data is sent to the CNN structured layer, and numeric input is sent to the fully connected layer separately.

3.4.2 Convolutional neural network

3D shape information is processed as an input for the convolutional neural network structure. The structure for CNN follows VoxNet suggested in [15] as it is fast and accurate for its complexity of the structure. The layers of the CNN structure are convolutional layers and a max-pooling layer.

Convolutional layer This layer receives a 3D array of voxel occupancy. In the first convolutional layer, 32 filters of size $5 \times 5 \times 5$ are applied to the 3D array with a stride of 2. The stride means how many voxels the filter is moved at a time during the convolution operation. In other words, the filter is shifted by 2 voxels in the first layer. This reduces the size of input data for the next layer. The network learns the filters that look for the specific patterns in the 3D shape classification resulting in feature maps as an output. The second convolutional layer has 32 filters of size $3 \times 3 \times 3$ with stride 1. The convolutional layers use a rectified linear unit (ReLU) as an activation function. ReLU is used because of its fast convergence [31].

Max-pooling layer Max-pooling with downsampling is used in the network to reduce dimensionality. The max-pooling takes the maximum value in the 2×2 size of filters from the feature map. Downsampling the feature map reduces parameters, and improves performance, and reduces overfitting.

3.4.3 Fully connected layer

In the CNN structure, the fully connected layer flattens the outputs of the pooling layer so that it can be merged with numeric inputs. This has 128 outputs that are connected to all neurons from the previous layer. The fully connected layer for numeric inputs computes matrix multiplications as in a regular neural network. The last output layer after concatenating two fully connected layers is also a fully

connected layer that calculates the classification scores. The last output layer has a softmax function which outputs a list of the probability of each category in the multi-class classification problem.

3.5 Validation

The validation of the trained model is done by comparing the suggested categories and the true categories of unclassified BIM elements. F_1-score is used to evaluate the prediction accuracy of the classification model. Performance is calculated using F_1-score which is the harmonic mean of precision and recall. Since the problem is multi-class classification, weighted F_1-score is used.

Precision In a classification problem, the precision of a category tells how many elements are predicted as the category are in the category. Precision is calculated from the number of true positives (TP) and false positives (FP). TP is the number of elements that are correctly predicted as a target category. FP is the number of elements that are predicted as an element of the target category, but the actual category of which are not the target category. Better models have higher precision.

$$Precision = TP/(TP + FP) \tag{5}$$

Recall This metric tells how many elements that are in the category are predicted as the category. Recall is calculated from the number of TP and false negatives (FN). FN is the number of elements that are predicted as an element of a non-target category, but whose actual category is the target category. Better models have higher recalls.

$$Recall = TP/(TP + FN) \tag{6}$$

F_1-score There is a trade-off between recall and precision such that the model cannot have both 100% precision and 100% recall. Therefore F_1-score is used as the harmonic mean of recall and precision. Weighted F_1-score is an average of F_1-score in all categories weighted by the number of elements (support) in each category.

$$F_1 = 2 \times Precision \times Recall/(Precision + Recall) \tag{7}$$

3.6 Prediction

The trained neural network model predicts how likely an unclassified element could be in each category. The category with the highest prediction score is suggested to users as a result.

4 Prototype

The prototype system consists of three components: data storage, notification service, and classification engine as shown in Figure 6. It is deployed on Amazon Web Services (AWS).

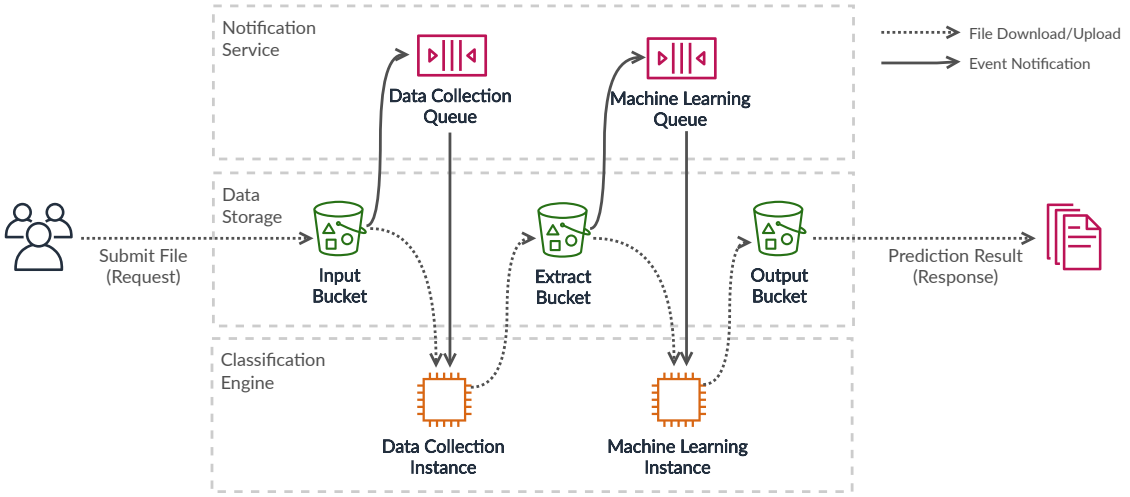


Figure 6: Prototype deployment

4.1 Data storage

All files and data are stored in the data storage component. The data storage is deployed in Amazon Simple Storage Service (S3) which is a cloud data storage and retrieval service with storage management features [32]. In S3, files are contained as S3 objects in S3 buckets. The prototype system has three S3 buckets to store different types of files from the system architecture.

Input bucket contains BIM files that are submitted to the system. These files are original IFC files or SMC files that have classified and unclassified building elements.

Extract bucket is intermediate storage for the extracted data of building elements that need to be retrieved for classification model training and prediction. Extracted data includes 3D shape and size data from geometry information, textual descriptions from semantic information and classification mapping information.

Output bucket is the place where the files with prediction results are stored. The prediction files are retrieved from this bucket.

4.2 Notification service

This component controls the execution ordering by emitting notification events between the data storage and the classification engine. The notification service is deployed in Amazon Simple Queue Service (SQS), and SQS offers a queue system that enables communication through messages. Two message queues in SQS are created to manage the data collection job and the machine learning job.

When a BIM file is uploaded to the input bucket, a message for the data collection instance is delivered to the queue. Then, the data collection instance executes a new job.

Similarly, the machine learning queue adds an item when the extract bucket receives file upload events, and the queue delivers the event message to the machine learning instance. The notification system makes sure that instances in the classification engine execute one job at a time in the pipeline.

4.3 Classification engine

The classification engine component consists of two sub-components, data collection component and machine learning component. Both are deployed as Amazon Elastic Compute Cloud (EC2) servers [33]. In the prototype system, there is one instance of each for the sub-components.

The data collection application, Solibri software, is installed in the data collection instance, and a data collection job is triggered when there is an unprocessed job in the data collection queue.

The machine learning instance is configured to perform neural network training and prediction based on the data that are extracted by the data collection instance when the machine learning queue has job items. When training and prediction are done, the instance uploads a result file back to the S3 output bucket so that it can be retrieved. Internally, EC2 instances keep polling the queue until there are unhandled messages and trigger the jobs one by one.

The data collection component is implemented with Solibri software. The machine learning component is implemented with Keras and Scikit-learn library in Python. Keras is a deep learning library written in Python [34]. It is a high-level API that enables easy and fast implementation of neural network models. Also, it supports running both on Central Processing Units (CPUs) and Graphics Processing Units (GPUs). Scikit-learn is a machine learning library for Python [35]. It integrates a wide range of machine learning algorithms.

4.4 Integration

The proposed classification system can be integrated with the existing classification tools in Solibri software product. The input file can be uploaded to the S3 input bucket to trigger the data collection and machine learning jobs. The resulting file that contains object ID and its category mapping suggestion will be displayed in Solibri software. Solibri classification shows the result in two big categories, *Classified* and *Unclassified* in the current system. BIM elements that can be assigned to each category are included in the *Classified* group, and other components all go to the *Unclassified* group. Based on the automatic classification engine's suggestions, a new group *Auto-classified* can be created and reviewed by the user. The user can accept the result of the automatic classification, or the user can reject and perform classification manually. In the early phase of the implementation, this user feedback about whether the classification satisfies the customer or not can be used to monitor

the accuracy and further improve the performance of the machine learning algorithms in the proposed classification system.

5 Experiments

In this section, two different sets of experiments are designed to evaluate the implemented system architecture from Section 3. The first experiment setting presents the data quality metrics and studies the effect of quality of data on system performance. The second experiment setting compares the system performance on different computing capacities of the system. The two factors in experiment settings, quality of data and system computing capacity, are closely related to classification performance. Therefore, analysis of their relationships and trade-offs can suggest how to adjust the factors to optimize the system and achieve the target system performance under the constraints each BIM model has.

5.1 Experiment setting 1

The purpose of this experiment setting is to understand how system performance is changed according to the quality of data. In this thesis, the following data quality metrics that reflect the characteristics of a BIM model are introduced: model size, data source, data completeness, and the number of categories. The definition and detailed experiment setting for each metric are given in this section.

5.1.1 Effect of model size on execution time

The model size is measured by the number of building elements in the BIM model. This metric is closely related to the execution time during classification. The execution time is measured by data collection time (`DC time`) and machine learning time (`ML time`). How `DC time` and `ML time` are measured will be explained in Section 5.4 in detail. When the amount of data is large, classification tasks require more time to process the data in general. This experiment measures the model size of 146 BIM models and compares how the execution time of each phase of classification changes depending on the model size.

5.1.2 Effect of data source on prediction accuracy

The data source metric tells whether the classification information in a BIM model is trustworthy or not. The quality of classification information directly influences the performance of the machine learning system, because the system trains the classification model based on the data on how each building element is classified. If a BIM model feeds wrong data into the machine learning system, it pollutes the input data and the classification system builds a low-quality classification model as a result. Since the quality of BIM models that are used in the experiments are not validated, there is a chance of collecting BIM models with not only accurate data but also some invalid data. BIM models could have invalid, incorrect, or irrelevant data because of the following reasons.

- Users lack understanding of the classification. Thus, some building elements are categorized into wrong groups.

- Some classification information is added only for test use. For example, a user might want to try out how they can add classifications to their BIM models.
- Some BIM tools automatically add classifications to BIM models when the file is imported or exported. In this case, users often leave the classification even when they do not use or maintain them.

To evaluate the performance of the proposed classification system, this experiment considers if the classification information is generated and maintained by a human or if it is from unknown sources. However, sources from where the data were generated in the first place are undefined in BIM files. Instead, a metric is defined to be *Well maintained by human* when the result of classification mapping is used for filtering the building elements in BIM, and *Unknown* otherwise. For example, the classification result is used in Rule Checking or Information Takeoff in Solibri software. Rule Checking is one of the main functionalities in Solibri software which examines problematic issues or requirements that the model should satisfy. Classification is done before rule checking so that the category can be used as a filter. For example, to make sure all doors have names, the checking rule can be applied only on elements that are classified as doors. Information Takeoff (ITO) is a powerful tool for collecting and reporting all the information in a BIM model. Using classification, ITO groups components and quantifies them. This information can be used for cost estimation or scheduling. It can be assumed that if the classification information is used or mentioned as a reference in any of these functionalities in SMC file format, the classification information is at least maintained by users and the content is trustworthy. This experiment compares the relationship between the data source metric and prediction accuracy.

5.1.3 Effect of data completeness on prediction accuracy

Data completeness means whether all 3D shape, numeric information, textual information is available in a BIM model. This thesis targets to utilize all different types of data in a BIM model. However, some building elements have only geometry information without any numeric or text attributes, or with only a limited number of attributes. The data completeness metric indicates the quality of data not only according to whether or not the BIM model contains only geometry of the building elements, but also according to whether it has descriptions on additional size, location, and text. This experiment examines the effect of data completeness in BIM. The experiment compares the prediction accuracy when all three types of data are available and when only the limited one or two types of data are available.

5.1.4 Effect of number of categories on prediction accuracy

The number of categories is considered when the classification has hierarchical levels. For example, in Figure 7, building elements are categorized into three major categories: furniture, kitchen, and door. In some cases, however, these categories can be divided further into multiple sub-categories, for instance, *sofa*, *desk*, *chair*

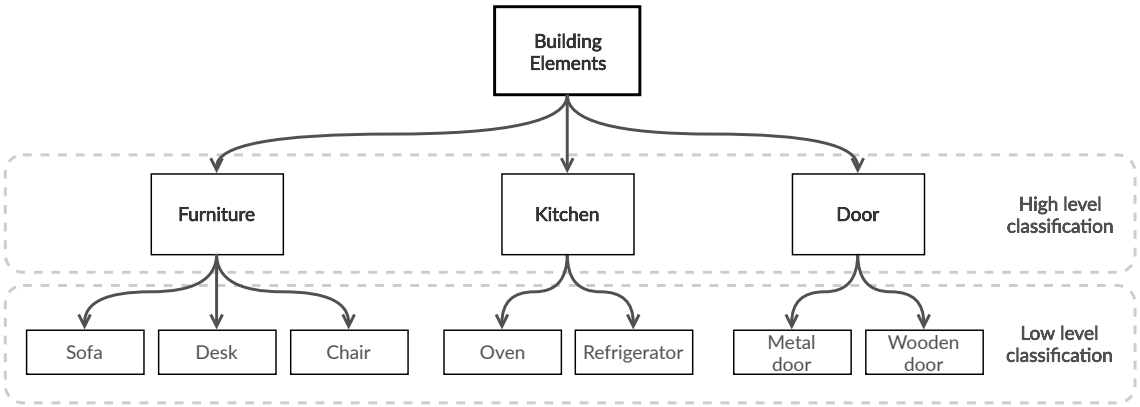


Figure 7: Example of different classification levels

in the *furniture* category. In this case, one element can have two labels which are *Furniture* in high-level classification and *sofa* in low-level classification. Both are meaningful depending on the situation where the classification is used. Generally, the high-level classification has fewer categories compared to the low-level classification, and this experiment compares the result of prediction accuracy in two different levels of classification.

5.2 Experiment setting 2

The purpose of experiment setting 2 is to study relationships between system configuration and performance in the classification system. In other words, the experiment compares the execution time of each component during classification in different environments.

There are two different types of configuration settings that are adjustable in the system: feature size and machine computation power. As the classification system extracts the data from the original BIM model, it is possible to decide how much data the system will extract during the data collection phase or the feature selection phase. Feature size configuration changes the number of features that the system uses for classification model training. Likewise, machine computation power is an adjustable configuration. This experiment examines the effect of different computing capacity on classification performance. Moreover, the prototype is deployed to virtual machines in a cloud environment. On cloud systems, it is easier and cheaper to control different computing resources, because the machines can be launched with as much capacity as the user needs without large investment or maintenance for a data center [36]. For example, it is possible to scale the system vertically by adding more computational power such as CPUs and GPUs depending on demand. Therefore, it is easier to test end-to-end performance on different computing machines for this experimental setting.

Due to time limitations, the experiments were done with 13 files randomly chosen from the BIM models from the experiment setting 1.

5.2.1 Effect of feature size on prediction accuracy and execution time

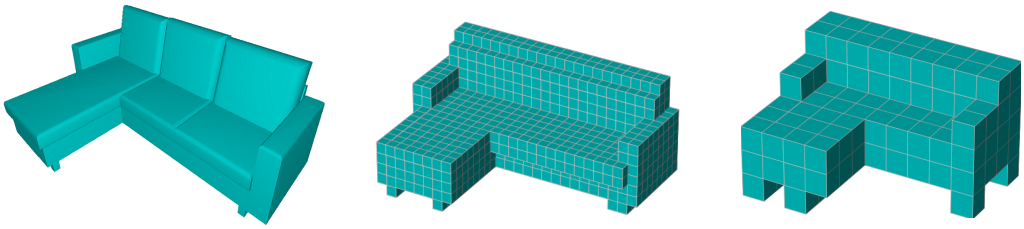
Generally, the number of input features can affect performance and overall accuracy in machine learning systems. In this experiment, the effect of the number of input features is studied. As input data is extracted from the original BIM file by the data collection component, the resolution of input data can be adjusted. Among the three different types of data (geometric shape, geometric size, and text descriptions), different representations of geometric shape and text description will be examined.

Metrics for input size are dimensions of the voxel grid and the size of the dictionary. The dimension of a voxel grid is the resolution of geometry information which influences data collection time, machine learning time, and prediction accuracy.

The dictionary size means the number of terms that are extracted when the bag-of-words model is created. This will not affect the data collection time because the keyword selection is done at pre-processing stage, not data collection stage. However, dictionary size can affect machine learning time which is spent in the machine learning component, and affect the accuracy of classification.

3D resolution Data resolution means how accurately the 3D representation is extracted from the original shape. When the size of the 3D grid is bigger, it can contain more information. Therefore, the 3D grid can have a smoother and much more accurate representation. How 3D shapes are changed in different sizes of grid space is shown in Figure 8. The resolution can be configured in the data collection component setting as a metric that indicates the quality of the extracted data. As this metric is adjustable, analysis of accuracy-time trade-off is done in this experiment.

Dictionary size Dictionary size means how many keywords from the bag-of-words model are used as input data. During the data pre-processing phase, each term is ranked by its importance in the classification task and only a fixed number of useful terms is selected. Like the 3D resolution metric, this metric is configured in advance to executing the system. Accuracy-time trade-off is analyzed by adjusting this metric.



(a) Original 3D shape (b) 24x24x24 voxel grid (c) 9x9x9 voxel grid

Figure 8: Examples of different 3D resolution

5.2.2 Effect of machine configuration on prediction accuracy and execution time

The performance of the classification system varies depending on the underlying computing capabilities, particularly when it comes to data collection which is done in Solibri software utilizing parallel computation in Java. The Keras backend and Tensorflow in the machine learning component also utilize the GPU to optimize the calculations. To examine the impact of the system capacity, the classification system executes data collection and machine learning in two different machines. This experiment is designed to compare the system performance in different machines with different numbers of CPU cores and GPUs during the data collection and machine learning phases.

Number of CPU threads Data collection is implemented to utilize multi-threading in Java. This experiment compares the performance of data collection with different numbers of CPU threads.

Availability of GPU Since computations in deep learning can be accelerated with GPUs, two different types of machine learning instances are launched: machines with only a CPU and machines with a CPU and a GPU.

5.3 Experiment data preparation

To validate the performance of the proposed classification system, BIM models that include one or more classifications are used for training. Classifications with only one category or categories without any classified elements are not considered.

To automate the training of classifiers for each classification, classified elements from each category are split into two groups, a training set and a test set, according to the following rules. Since the actual categories of unclassified elements are not available, it is impossible to verify the result manually. Therefore, the experiments consider only the elements that are classified in the BIM files. For training, the classified elements are divided into both training data and test data. When dividing the classified elements into training data and test data, the output label of test data is not used as training so that the label information can be used to validate the proposed classification system. The existing classification information is added by either using the designing tool, rule definition or manually. The pre-classified building elements are randomly split into 70% used for training and 30% used for testing. Since classified elements in a category are used for both training and testing, at least two classified elements are required for each category.

There is one more extra pre-processing step for this mock test data. If the classification is done automatically by Solibri software, the classification is already done in original building models by rules. This could contain the keywords that the rules require. This means that the building elements might contain the keywords in their properties and these keywords can affect the performance of the result. Especially rule-defined components are classified already according to the fields that

are defined in rules. For example, if there is a rule that classifies components with a name *Slab* as *Ceiling* category, the keyword *Slab* should be removed from the mock test data to assume that the components cannot be classified by the rule and remained as unclassified after rule-classification in Solibri software. To evaluate if the machine learning classifier predicts accurately without the pre-defined keywords in the rules, all the keywords that are mentioned in the rule are removed from the target properties.

5.4 Performance measurement

The experiment compares system performance in different settings. The performance means the accuracy of the classification and the execution time of the system. The accuracy is measured by F_1-score, and the execution time is measured in seconds. Specifically, data collection time and machine learning time are measured. The details of the execution time in each component are presented in Figure 9.

5.5 Environments

Each component in the prototype system is configured in different environments for the different purposes of the experiments. Environment A (Env A) is a local machine that is used for development and testing the system. Batch execution for comparing the result of multiple BIM files is done in this environment. The CPU in this machine has 6 cores and 12 threads. Experiment setting 1 is done in Environment A because the goal of this experiment is to understand the quality of data and how it changes the system performance in one machine.

Environment B (Env B) and Environment C (Env C) are deployed EC2 instances on AWS as prototype design. Different instance types for each component are configured to compare the performance of the system in different environments. Env B and Env C have different settings for the data collection component and the machine learning component. Thread counts for AWS EC2 instances are converted from Virtual Central Processing Unit (vCPU) count since vCPU is the number of CPU cores multiplied by threads per core [37]. In Env B, data collection is on a r5.large EC2 instance with 2 vCPUs and machine learning instance is on a r5.xlarge instance with 4 vCPUs. R5 instances are suitable for memory-intensive tasks. Env C has more compute capacity for both components, which use a r5.xlarge instance for data collection and a g4dn.xlarge instance with a GPU [38] for machine learning. Performance of the data collection component can be compared between two different settings in CPU threads and memory, and can be examined on a machine without a GPU and on a machine with a GPU. These two environments are used for the experiment setting 2. Details on the environments are listed in Table 1.

The experiments in this study are performed in three different environments. The experiments do not consider all possible combinations of the local machine or EC2 instances. The experiments could be extended to use many different steps of computation powers, for instance, using 10 different environments with 1 to 10 CPU threads. However, for simplicity the scope of the experiments is limited to run in two

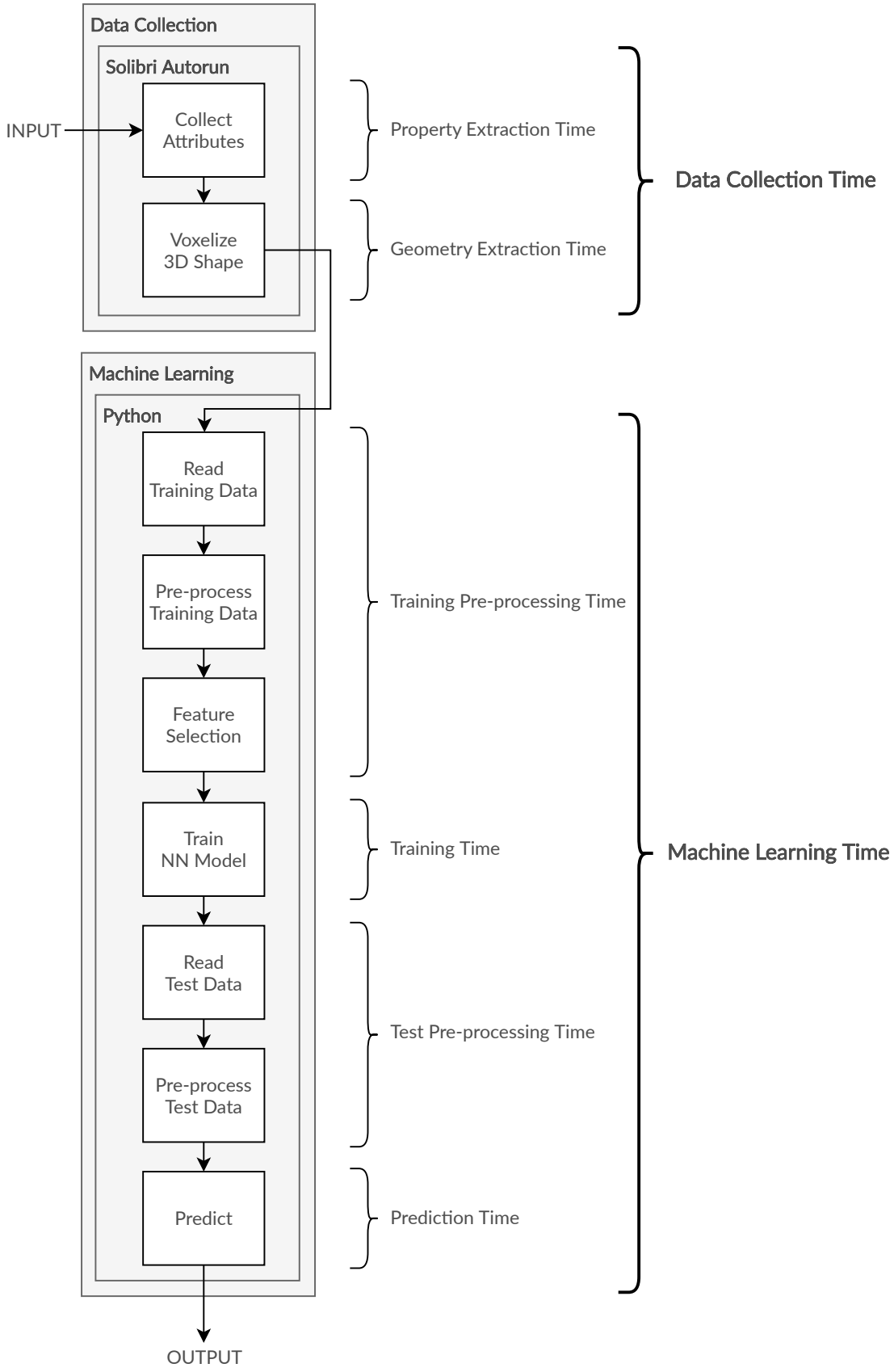


Figure 9: Tasks in data collection time and machine learning time

	Platform	Component	Threads	Memory	GPU
Env A	Local Machine	All	12	16	Nvidia GTX 1660
Env B	AWS	Data Collection	2	16	N/A
		Machine Learning	4	32	N/A
Env C	AWS	Data Collection	4	32	N/A
		Machine Learning	4	16	Nvidia T4

Table 1: Environment list and specification for the data collection component and machine learning component

environments which are enough to provide sufficient data to observe the potential factors and trends of the behavior of the system.

6 Results and Discussion

This section presents the results of the experiments that are carried out to implement the ML-based classification system for BIM and find relationships between data quality, system configuration, and system performance. Section 6.1 shows the effect of the characteristics of a BIM model on classification performance using data quality metrics that are defined in the experiment setting in Section 5.1. Section 6.2 evaluates how the system configuration in the experiment setting in Section 5.2 affects the classification performance, as the configuration can be adjusted.

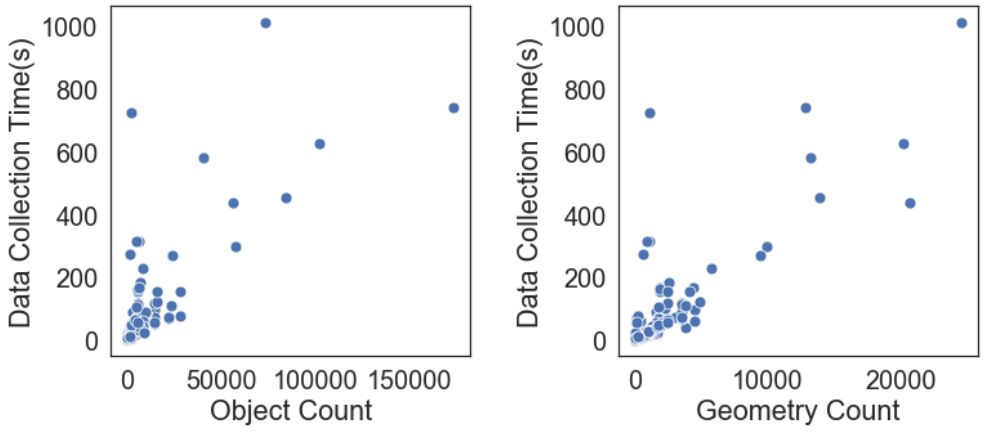
6.1 Data quality

To determine the performance changes of the implemented system, separate experiments were designed for each data quality metric. In this subsection, the results of the experiments are discussed in detail.

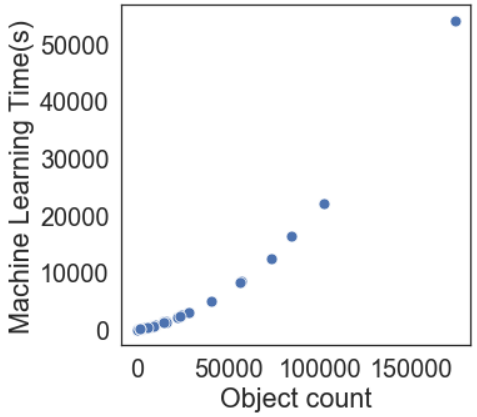
6.1.1 Effect of model size on execution time

This experiment was done using 176 BIM models that are selected from sample BIM models. Figure 10 shows the system performance against the model size metric. The model size is defined as the total number of objects in a BIM model. As can be seen in Figure 10 (a), data collection time which measures the execution time in the data collection phase shows an increasing trend since there are more objects in the BIM model. However, the result does not show a linear relation between object count and data collection time. There are reasons for this behavior which could be explained by understanding how the voxelization works in data collection. In reality, total execution time in the data collection phase mostly comes from geometry extraction. Therefore, when there are many elements to voxelize, more time is required for data collection. However, voxelization is not executed for every element because many elements share a common geometry representation in a BIM model. The data collection time correlates directly with the number of different geometric representations in the model. Multiple elements with identical geometric representation but with different rotation can be processed as fast as a model with one element only. Figure 10 (b) shows how data collection time changes against the number of geometries in a BIM model. The relationship between data collection time and geometry count is more linear than the relationship between data collection time and object count. Still, the geometry count alone cannot be used to predict the actual data collection time. One reason for this is that the voxelization time for one geometry varies depending on the complexity of the geometry shape. Voxelization requires more calculation steps and time when the geometry is for a sphere where the 3D voxel grid consists of different numbers of 1 and 0 compared to a simple cube which will be extracted as a 3D array that is filled with only 1.

Contrary to the relationship between model size and data collection time, machine learning time has a linear correlation to the model size. In Figure 10 (c), the most data points lie on the diagonal line of the graph. This is because each element is



(a) Object vs. `DC time` **(b)** Geometry vs. `DC time`



(c) Object vs. `ML time`

Figure 10: Impact of object count

considered as one input data instance with a fixed length of data for classification model training regardless of the complexity of the shape. This result demonstrates that model size can be one of the data quality metrics that help users predict the execution time of the classification system based on the object count in advance.

6.1.2 Effect of data source on prediction accuracy

Figure 11 shows the results of system validation for different data sources. There are 162 *Used* classifications and 236 *Unused* classifications. As a result, the BIM models that are considered to be maintained by the user (Used category) have better average accuracy compared to the BIM models for which the usage of the classification is unknown. The mean and median accuracy of unused classifications are 0.81 and 0.91, respectively. And mean and median accuracy of *Used* classifications are 0.90 and 0.98. Also, in Figure 11, the position of the interquartile range of the *Used* classification is closer to 1 compared to the range of *Unused* classification. This result indicates that the implemented system performs better with BIM models that have well-maintained data.

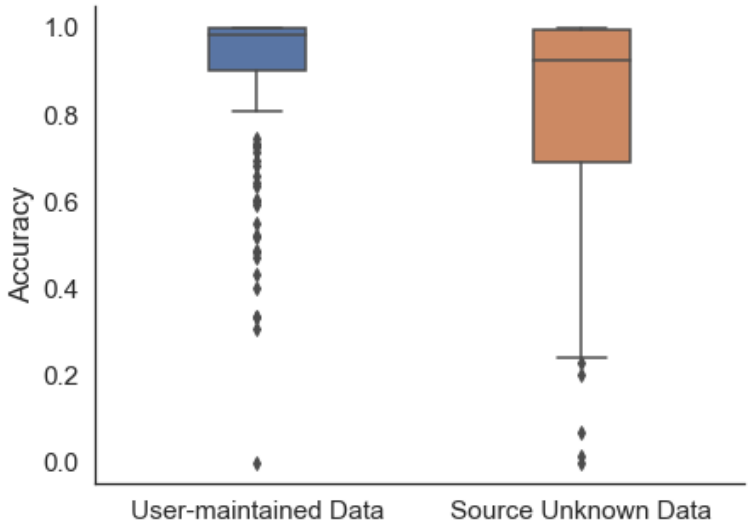


Figure 11: Average result of maintained classification and unknown classification

6.1.3 Effect of data completeness on prediction accuracy

Figure 12 shows that the model prediction accuracy is higher on average when many different types of data are combined. This experiment is done with 585 classification cases and the result of each case is shown in Figure 12. The training model performs better when both 3D shape information and size information are used compared to when only 3D shape information is used. Similarly, when the textual description information is used in addition to 3D and size information, the overall performance increases precipitously. When only shape information is used for classification, the average F_1-score is 0.72, and when shape information and size are used for classification, the F_1-score is 0.75. Finally, when all the information—3D shape,

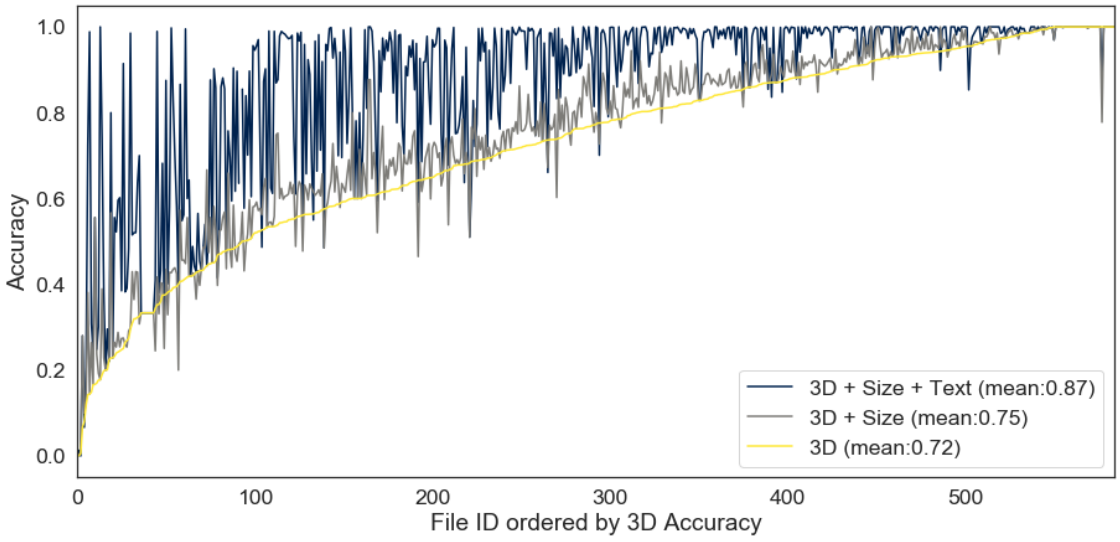


Figure 12: Accuracy improvements with different input data types

size, and text—are available, the accuracy F_1-score is improved up to an average of 0.87. This result shows that as more training data is available, the classification performance improves.

Let us investigate one test model as an example of the accuracy improvement. Confusion matrices for experiments with different input data types are shown in Figure 13. Confusion matrix is one effective way of visualizing the performance of multi-class classifications [39]. In a confusion matrix, each row represents the true category and each column represents the predicted category. For example, an element that is on row *Basic Wall* and column *Basic Wall* means the number of *Basic Wall* elements in the test data that are predicted as *Basic Wall*. Similarly, an element on row *Basic Wall* and column *Kitchen Sink* means the number of *Basic Wall* elements that are misclassified to *Kitchen Sink*. In other words, the confusion matrix gives information on true positives, true negatives, false positives, and false negatives in the classification result. Since there are many rows and columns in the experiment, the elements in the matrix are represented in different colors, not numbers. When the color is close to dark blue, it means most of the elements in this category are classified into the correct category. When the color is lighter, the trained classifier could not guess the true label for the elements in the category. When the overall accuracy of prediction is high, a confusion matrix has dark blue elements in the diagonal of the matrix since the order of categories in the row and the column are the same.

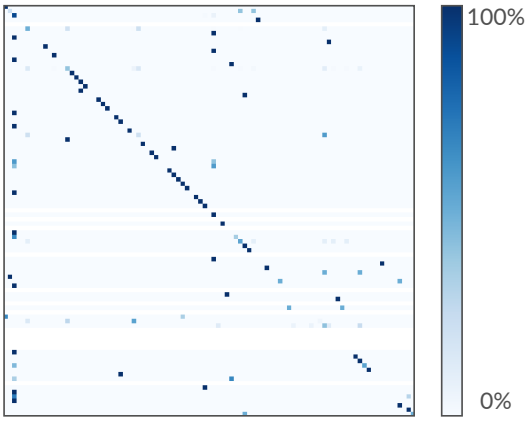
This result shows that the prediction accuracy improves when the classification model can use all 3D shapes, numeric, and text attributes. Different types of data describe different aspects of the elements.

6.1.4 Effect of number of categories on prediction accuracy

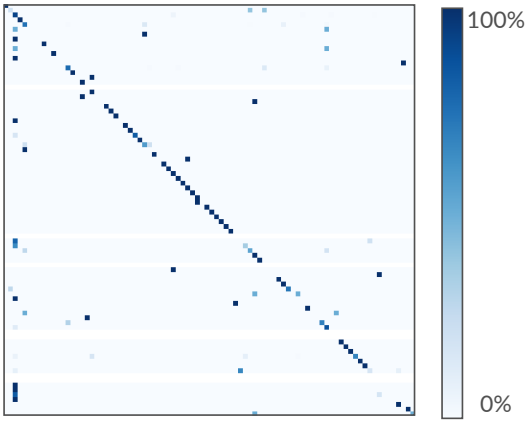
In order to find the effect of the number of categories on prediction accuracy, two BIM models, Model A and Model B that share the same classification rule, were used in this experiment.

In this experiment, all the labels that are assigned manually to the elements in Model B were removed, and the machine learning trained classifier assigns new labels based on the categories in Model A. In Model A, classification was done in Level 1 and Level 2 that have a hierarchical difference. Level 1 classification categorizes elements at a higher level and Level 2 categorizes the elements into more specific categories using details of the elements. For example, Level 1 category *Furniture* can have two Level 2 categories, *Furniture:chair* and *Furniture:desk*. These two Level 2 categories are meant to classify *Furniture*, but they are for different types of furniture more specifically. In Table 2, the number of categories was described in different classification levels in Model A and Model B.

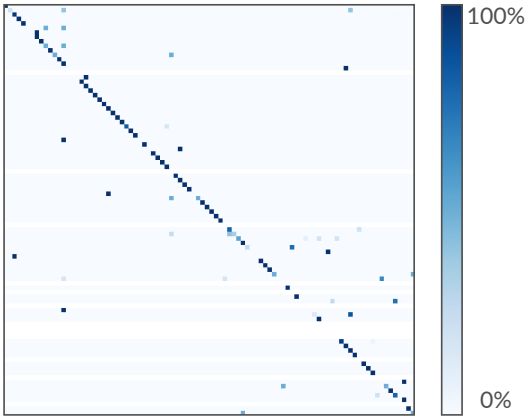
The result of this experiment is shown in Figure 14 and Table 3. The prediction accuracy is always higher when there are a smaller number of categories in BIM models regardless of the available data types. With 3D shape, numeric, and text information all available, Level 1 classification was trained with Model A to predict 194 categories and achieved 99% accuracy in predicting 79 categories in Model B,



(a) 3D



(b) 3D + Size



(c) 3D + Size + Text

Figure 13: Confusion matrices in an example BIM model

Level	Model A	Model B
Level 1	194	79
Level 2	418	84

Table 2: Number of categories in different classification levels

whereas Level 2 classification was trained to predict 418 categories with Model A and achieved 73% accuracy in predicting 84 categories in Model B.

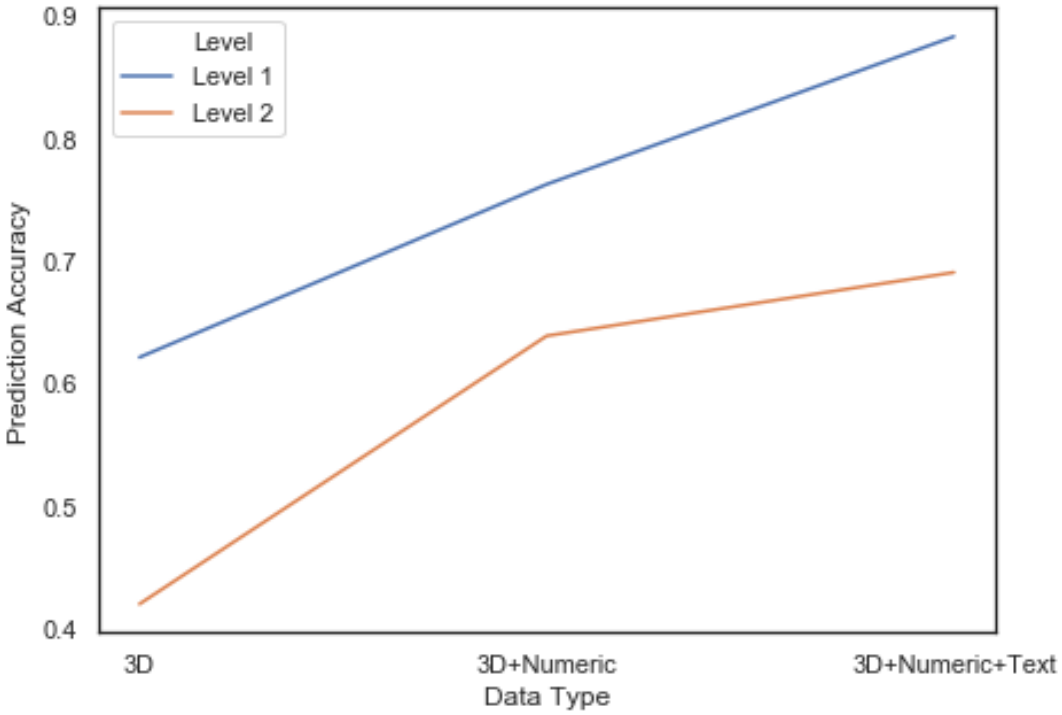


Figure 14: Accuracy in different levels of classification

Data Type	Level 1	Level 2
3D	62.12	42.00
3D+Numeric	76.23	63.87
3D+Numeric+Text	99.28	73.11

Table 3: Prediction accuracy (%) in different classification levels and data types

6.2 System configuration

The experiment results in this subsection are provided to determine the effect of feature size and machine computing power on classification performance. As system configuration is adjustable, appropriate strategies of system settings are given as a trade-off analysis. The experiments are done using 13 files as described in Table 4.

File ID	Number of Objects	Number of Geometries	Number of Categories
File1	1718	943	110
File2	2384	1153	242
File3	2236	989	79
File4	2870	1196	179
File5	6651	1124	39
File6	1570	85	15
File7	299	102	2
File8	1490	756	19
File9	22278	3531	23
File10	601	118	29
File11	487	303	20
File12	6166	1125	51
File13	6210	2205	5

Table 4: File list and quality metrics

6.2.1 Feature size

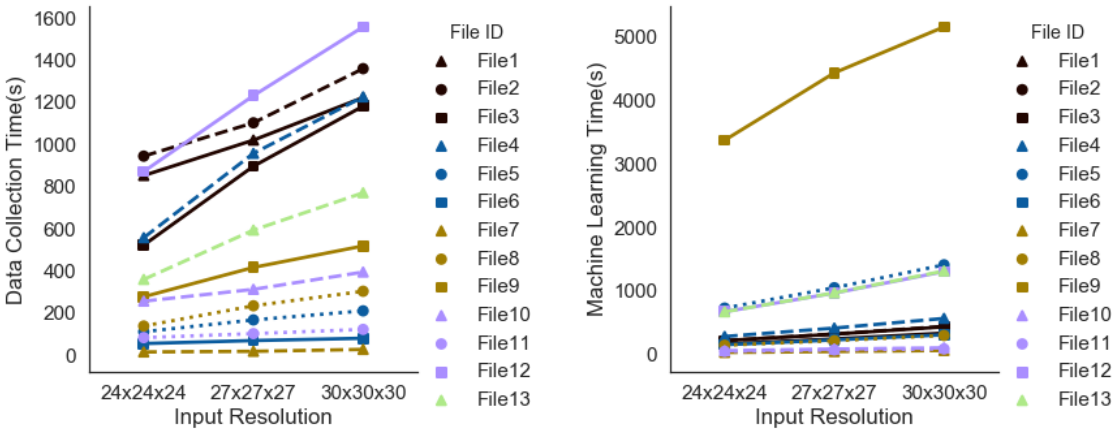
Feature size is the number of attributes that are used to represent the BIM element. The system controls feature size by adjusting the 3D resolution and dictionary size.

3D resolution In Figure 15, it is shown that high-resolution input data extraction requires more time for both data collection and the machine learning phase. Data collection time is highly related to the number of geometries in the BIM file. Even when there are many elements in one file, the elements can share the geometric information, which means that the 100 different elements can have the same shape. In this case, the data collection time is calculated only for one geometry, not 100 geometries. As shown in Figure 15 (a), files with higher data collection time, such as Files 1–4 and File 12, contain more geometries for their number of objects. On the other hand, Files 5, 9, and 13 also have many geometries, but the data collection time is notably low for these files. This may have taken place due to the complexity of the shapes being low in these files. When the data collection component voxelizes the object shape, it takes more time for the objects with a more complex shape and more polygons. On the contrary, machine learning time is directly related to the number of elements in Figure 15 (b). This is because the input data rows are created for each element even if multiple elements share the same geometry. In machine learning algorithms, the input rows with duplicated data are often removed. However, the elements with the same geometry do not necessarily have the same data in BIM models since the elements with the same geometric shape can have a different size or properties. There are some exceptions where the accuracy fluctuates inconsistently. However, the average accuracy of the classifier increases as the voxelized 3D object represents the original object more accurately, as shown in Figure 15 (c).

Dictionary size Figure 16 (b) shows that the average accuracy tends to increase as the dictionary size increases. However, the total time for processing a bigger dictionary does not seem to be increased significantly in Figure 16 (a). The reason can be that the `ML time` is related to the total keyword count in a BIM model, not the selected keyword count. Especially, the result for File 9 shows significantly high `ML time` in Figure 16 (c). This is because the total number of terms in the file is much bigger than in other files. It also requires more time for selecting useful and important keywords out of all the terms during the pre-processing phase.

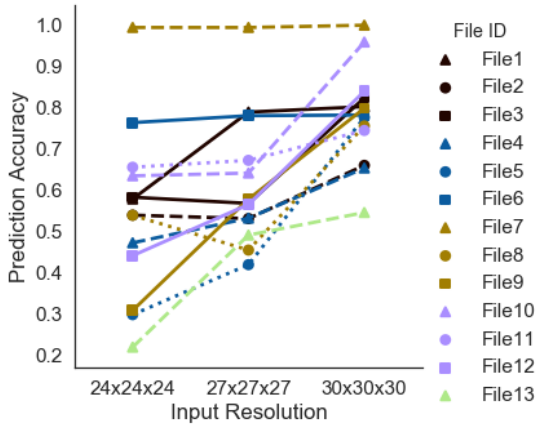
6.2.2 Machine computing power

The classification system was experimented with in two different system environments, Environment B (Env B) and Environment C (Env C). In general, Env C has a more powerful computing capacity compared to Env B for both the data collection component and the machine learning component.



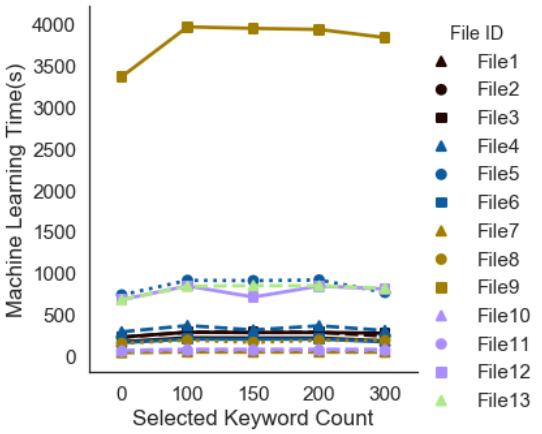
(a) Impact of input resolution on `DC time`

(b) Impact of input resolution on `ML time`

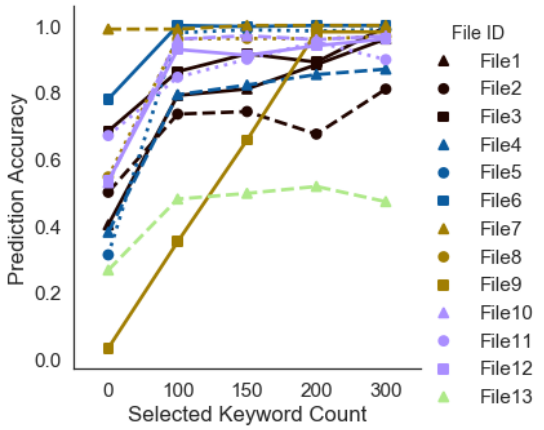


(c) Impact of input resolution on prediction accuracy

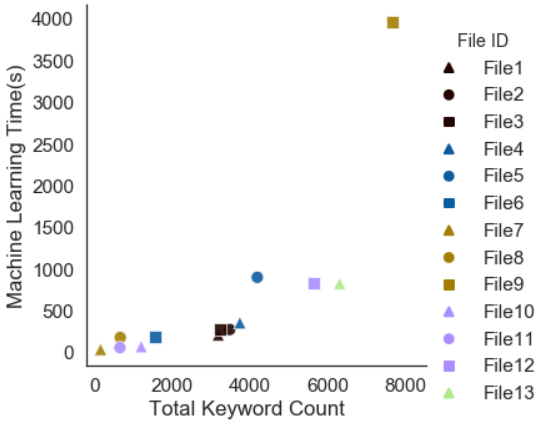
Figure 15: Impact of different 3D voxel grid resolutions



(a) Impact of selected keyword count on `ML time`



(b) Impact of selected keyword count on accuracy



(c) Impact of total keyword count on `ML time`

Figure 16: Impact of different dictionary sizes

CPU In Figure 17 (a), the overall performance was improved in Environment C compared to Environment B. Data collection is related to the number of threads in the CPU since the extraction is executed in parallel. As the thread number is doubled, the average `DC time` is reduced to around 55% as Table 5 represents.

GPU It is shown in Figure 17 (b) that the average machine learning time seems to have more dramatic improvements in Environment C. In Table 6, the average `ML time` in Environment C was around 20% of the time spent in Environment B. However, the improvement in File 9, which is only 42%, was not as significant as in other files. This is because pre-processing in the ML phase does not utilize GPU. Therefore, as there are more elements in the file, the time for pre-processing increases which leads to a long overall `ML time`.

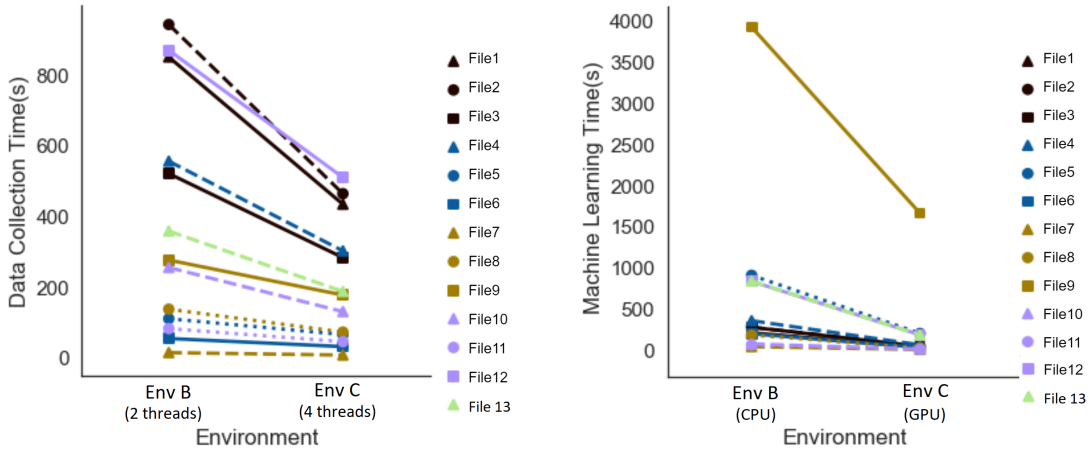
File ID	Env B	Env C	Improvement (%)
File1	852	435	51.05
File2	944	465	49.25
File3	522	284	54.40
File4	557	303	54.39
File5	111	66	59.45
File6	55	32	58.18
File7	15	8	53.33
File8	138	73	52.89
File9	277	178	64.25
File10	256	131	51.17
File11	83	46	55.42
File12	871	511	58.66
File13	359	188	52.36
Average			54.99

Table 5: Improvements of `DC time` per file

6.3 Recommendation

This subsection provides several strategies for system and feature size configuration based on the quality of data analysis. It helps recommend practical system configurations for different situations and constraints.

One constraint in the ML-based classification system is that it should not take a too long time compared to the current classification tools. The users would consider a total execution time of more than 10 minutes to be unacceptable based on Solibri user feedback. As discussed in the previous subsections, large-sized BIM models tend to have longer execution times. One sample BIM model with the highest 25% of the model size from the experiment results in Section 6.1.1 is an example of a large-sized



(a) Data collection time comparison **(b)** Machine learning time comparison

Figure 17: Impact of different compute capacities

File ID	Env B	Env C	Improvement (%)
File1	210	34	15.99
File2	279	48	17.42
File3	278	45	16.30
File4	358	63	17.76
File5	911	202	22.22
File6	198	31	15.65
File7	41	7	16.48
File8	183	26	13.98
File9	3934	1668	42.40
File10	75	12	15.87
File11	63	10	16.55
File12	837	185	22.16
File13	840	184	21.92
Average			19.59

Table 6: Improvements of `ML time` per file

BIM model. This BIM model has 5,891 elements and 3,480 geometry representations and it spent 116 seconds (~2 minutes) for extracting the 3D shape in $24 \times 24 \times 24$ resolution and all numeric and text attributes during the data collection phase. Its machine learning time was 416 seconds (~7 minutes). The total execution time is around 9 minutes in Environment A with 12 CPU threads and a GPU as stated in Table 1.

Let us take a look at one more BIM model as an example, the one with the highest model size from the experiment in Section 6.1.1. This model has 173,210 elements and 12,855 geometry representations. For this model, data collection was done in 744 seconds (~12 minutes) and the machine learning phase took 54,074 seconds (~15 hours). In real life, spending 15 hours on classification is unreasonable. In this case, it is recommended to use a lower 3D shape resolution, such as $16 \times 16 \times 16$. This will reduce the feature size of 3D shape data from $24 \times 24 \times 24 \times 173,210 = 2,394,455,040$ to $12 \times 12 \times 12 \times 173,210 = 299,306,880$, which will be similar to the feature size when the model size is 21,651 with configured 3D resolution of 24 ($24 \times 24 \times 24 \times 21651 = 299,306,880$). Although the actual execution time can vary depending on the other characteristics of the BIM model, a BIM model with 21,651 elements would spend around 100 seconds (~2 minutes) for data collection and around 2,000 seconds (~33 minutes) for machine learning according to the experiment result in Section 6.1.1, which would be a substantial improvement. This is still out of the acceptance range from the user's perspective, nevertheless, using a more powerful GPU or multiple GPUs will accelerate the performance further. As a trade-off, however, it is shown that the lower 3D data resolution yields a lower accuracy of the classification result. As the system compresses the 3D shape data and loses some of the information from the original 3D shape, adjusting the dictionary size to select as many keywords as possible will compensate for the lower accuracy that might be caused by lower 3D data resolution.

7 Conclusion and Future Work

This section reviews the aim of the thesis and the solution that was introduced to apply ML to a BIM element classification system. This section also briefly summarizes the experiment results on data quality analysis in the proposed ML-based classification system and its importance on the practical application of the study. Furthermore, the result is evaluated, and potential improvements are discussed.

7.1 Conclusion

The goal of this thesis was to improve the existing classification tools for BIM elements. From the experiment, it was found that the rich information of BIM, which consists of two main parts: geometry and semantic information, improves the accuracy of the classification system. Then, the suitable machine learning algorithms were adapted so that the system can process the chosen data: 3D shape, size and text information. It is empirically proven that the BIM data enables automatic classification that does not require human intervention, unlike the conventional classification tools that require users to either assign the correct categories or define the classification rules manually. Also, through the experiments, trade-offs between the quality of data, execution time and prediction accuracy were studied. This analysis of trade-offs was done in a controlled environment of the prototype which was deployed on a cloud system. The investigation showed that higher accuracy results require more time for execution. However, as more computing resources can be added to the system, it is possible to compensate for the increase in execution time by increasing the computing power.

Moreover, this thesis has studied the quality metrics of BIM models that characterize and represent the model. The metrics are the number of elements in the model, the 3D voxelization dimension size, and dictionary size. Experiment results showed that the accuracy of classification increases when using more accurate representations of the element. This means that when the system extracts data, which are close to the original element shape or description, the accuracy increases. However, it brings significant growth of elapsed time in the data collection and machine learning phases. Also, models with many elements tend to require more time to process the tasks, although the rate of how many elements share the same geometry should also be considered since data collection time is dependent on this factor. The experiments showed how adjusting input resolution and the number of input features can improve accuracy. On the other hand, there is a trade-off with the time spent during the classification task. In particular, the time-accuracy trade-off is much more dramatic in the 3D dimension size compared to the dictionary size. Therefore, it is recommended to adjust dictionary size first before trying to increase the dimensions of the 3D voxel grid.

Also, the prototype system was deployed in different machines to study the improvements in performance and accuracy. It is tested in two types of the environment on AWS, one machine with less compute capacity and one with more compute capacity in CPU, memory, and GPU. More powerful machines may be

costly, but provide remarkable improvements in the performance of the system. If there is a large number of elements in a BIM model to be classified, and it is too time-consuming, more powerful machines can come in handy.

7.2 Suggestions for improvements

Like all the other previous research on BIM element classification, this study has its own limitations that have been left as future work. In terms of the applied machine learning algorithm, a neural network technique with global settings is implemented and applied to all different BIM models. In the future, the system can be upgraded to automatically choose the suitable algorithm for each BIM model and optimize the machine learning training to improve accuracy. Similarly, the system can be automatically scaled vertically or horizontally depending on the characteristics of each BIM model. Lastly, the overall pipeline can be upgraded by utilizing tools and frameworks for machine learning, such as MLflow and Kubeflow for better management of the ML life cycle [40].

References

[1] A. Borrmann, M. König, C. Koch, J. Beetz, Building Information Modeling, Springer, 2018.

[2] S. Jones, D. Laquidara-Carr, A. Lorenz, B. Buckley, S. Barnett, The business value of bim for infrastructure 2017, SmartMarket Report (2017).

[3] Classification systems and their use in autodesk revit® managing the "i" in bim, https://www.biminteroperabilitytools.com/classificationmanager.php, [Accessed: 01.10.2019].

[4] Archicad 23 reference guide, https://helpcenter.graphisoft.com/user-guide/88835/, [Accessed: 08.02.2020].

[5] Solibri, http://www.solibri.com/, [Accessed: 15.09.2019].

[6] Solibri model checker help v9.9 classification settings dialog, https://solution.solibri.com/help/smc/9.9/en/Help.htm, [Accessed: 16.02.2020].

[7] I. ISO, 16739: 2013 industry foundation classes (ifc) for data sharing in the construction and facility management industries, International Organization for Standardization (2013).

[8] E. Alpaydin, Introduction to machine learning, MIT press, 2009.

[9] R. Schapire, Machine learning algorithms for classification, https://www.cs.princeton.edu/~schapire/talks/picasso-minicourse.pdf, [Accessed: 25.11.2019].

[10] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016, [Accessed: 01.02.2020].

[11] S. B. Kotsiantis, I. Zaharakis, P. Pintelas, Supervised machine learning: A review of classification techniques, Emerging artificial intelligence applications in computer engineering 160 (2007) 3–24.

[12] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, nature 521 (7553) (2015) 436–444.

[13] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, Backpropagation applied to handwritten zip code recognition, Neural computation 1 (4) (1989) 541–551.

[14] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3d shapenets: A deep representation for volumetric shapes, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2015, pp. 1912–1920.

[15] D. Maturana, S. Scherer, Voxnet: A 3d convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2015, pp. 922–928.

[16] H. Su, S. Maji, E. Kalogerakis, E. Learned-Miller, Multi-view convolutional neural networks for 3d shape recognition, in: Proceedings of the IEEE international conference on computer vision, 2015, pp. 945–953.

[17] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: A large-scale hierarchical image database, in: 2009 IEEE conference on computer vision and pattern recognition, Ieee, 2009, pp. 248–255.

[18] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, L. J. Guibas, Volumetric and multi-view cnns for object classification on 3d data, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 5648–5656.

[19] S. Zhi, Y. Liu, X. Li, Y. Guo, Lightnet: A lightweight 3d convolutional neural network for real-time 3d object recognition., in: 3DOR, 2017.

[20] J. Wu, J. Zhang, Automated bim object classification to support bim interoperability, in: Construction Research Congress 2018: Sustainable Design and Construction and Education, 2018, pp. 706–715.

[21] T. Krijnen, M. Tamke, Assessing implicit knowledge in bim models with machine learning, in: Modelling Behaviour, Springer, 2015, pp. 397–406.

[22] B. Koo, B. Shin, Applying novelty detection to identify model element to ifc class misclassifications on architectural and infrastructure building information models, Journal of Computational Design and Engineering 5 (4) (2018) 391–400.

[23] V. Stojanovic, M. Trapp, R. Richter, J. Döllner, A service-oriented approach for classifying 3d points clouds by example of office furniture classification, in: Proceedings of the 23rd International ACM Conference on 3D Web Technology, ACM, 2018, p. 2.

[24] The importance of bim in facilities management, https://fmlink.com/articles/the-importance-of-bim-in-facilities-management/, [Accessed: 26.11.2019].

[25] F. Lomio, R. Farinha, M. Laasonen, H. Huttunen, Classification of building information model (bim) structures with deep learning, in: 2018 7th European Workshop on Visual Information Processing (EUVIP), IEEE, 2018, pp. 1–6.

[26] R. P. Charette, H. E. Marshall, UNIFORMAT II elemental classification for building specifications, cost estimating, and cost analysis, US Department of Commerce, Technology Administration, National Institute of Standards and Technology, 1999.

[27] Omniclass®, https://www.csiresources.org/standards/omniclass, [Accessed: 26.11.2019].

[28] H. Schütze, C. D. Manning, P. Raghavan, Introduction to information retrieval, in: Proceedings of the international communication of association for computing machinery conference, Vol. 4, 2008.

[29] Y. Yang, J. O. Pedersen, A comparative study on feature selection in text categorization, in: Icml, Vol. 97, 1997, p. 35.

[30] D. Griffiths, J. Boehm, A review on deep learning techniques for 3d sensed data classification, Remote Sensing 11 (12) (2019) 1499.

[31] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in neural information processing systems, 2012, pp. 1097–1105.

[32] Amazon s3, https://aws.amazon.com/s3/, [Accessed: 25.12.2019].

[33] Amazon ec2, https://aws.amazon.com/ec2/, [Accessed: 25.12.2019].

[34] F. Chollet, et al., Keras, https://keras.io, [Accessed: 02.02.2020] (2015).

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[36] Six advantages of cloud computing, https://docs.aws.amazon.com/whitepapers/latest/aws-overview/six-advantages-of-cloud-computing.html, [Accessed: 01.02.2020].

[37] Optimizing cpu options, https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/instance-optimize-cpu.html, [Accessed: 15.11.2019].

[38] Amazon ec2 instance types, https://aws.amazon.com/ec2/instance-types/, [Accessed: 03.12.2019].

[39] E. Ameisen, Building Machine Learning Powered Applications: Going from Idea to Product, O'Reilly Media, Incorporated, 2020.
URL https://books.google.fi/books?id=-3mfxgEACAAJ

[40] R. Isdahl, O. E. Gundersen, Out-of-the-box reproducibility: A survey of machine learning platforms (2019).